Comparative Study of Chaos-Embedded Particle Swarm Optimization

Dongping Tian(✉), Bingchun Li, Chen Liu, Haiyan Li, and Ling Yuan

School of Computer Science and Technology, Kashi University, Xuefu Avenue, Kashi City 844006, Xinjiang, China
tdp211@163.com

Abstract. Particle swarm optimization (PSO) is a population-based stochastic search algorithm that has been widely used to solve many real-world problems. However, like other evolutionary algorithms, PSO also suffers from premature convergence and entrapment into local optima when addressing complex multimodal problems. In this paper, we propose a chaos-embedded particle swarm optimization algorithm (CEPSO). In CEPSO, the chaos-based swarm initialization is first applied to yield high-quality initial particles with better stability. Afterwards the chaotic inertia weight and the chaotic sequence based random numbers are introduced into the velocity update scheme for PSO to improve its global and local search capabilities. In addition, two different mutation strategies (chaos and levy) are utilized to enhance the swarm diversity without being trapped in local optima. Finally, the CEPSO proposed in this work is compared with several classical PSOs on a set of well-known benchmark functions. Experimental results show that CEPSO can achieve better performance compared to several other PSO variants in terms of the solution accuracy and convergence rate.

Keywords: Particle swarm optimization · Chaos · Swarm diversity · Inertial weight · Premature convergence · Local optima · Mutation strategy

1 Introduction

Particle swarm optimization (PSO) is a stochastic population-based method motivated by the intelligent collective behaviour of some animals such as flocks of birds and schools of fish [1]. Due to the advantages of PSO include fast convergence toward the global optimum, easy implementation and fewer parameters to adjust. All of these make it as a potential method to solve different optimization problems in a wide variety of applications, such as text mining [2], data clustering [3], image processing [4], optimal scheduling [5] and machine learning [6], etc. Meanwhile, an numerous number of different PSO variants have been proposed by researchers in the literature, and most of them can achieve encouraging results and impressive performance. However, PSO still suffers from the issues of premature convergence and entrapment into local optima like other stochastic search techniques, especially in the context of the complex multimodal

Published by Springer Nature Switzerland AG 2022
Z. Shi et al. (Eds.): IIP 2022, IFIP AICT 643, pp. 15–29, 2022.
https://doi.org/10.1007/978-3-031-03948-5_2

optimization problems. To tackle the above issues, a huge number of PSOs have been developed to enhance the performance. From the literature, these previous works can be roughly divided into the following categories: (i) swarm initialization. Note that in (i), some PSO variants are initialized with chaos sequence [4, 7], opposition-based learning [2, 8], and some other initialization strategies [9] instead of the purely random mechanism to improve PSO performance. In particular, the chaos based swarm initialization has been extensively studied in our prior works [10, 11] with significantly improved performance. (ii) Parameter selection. The parameters inertia weight [4], acceleration coefficients [10], and random numbers [2] attract much more attention and have become focus of research in the area of PSO in recent years. (iii) Non-parametric update. In this paradigm, there is no need to tune any algorithmic parameters in PSO by removing all the parameters from the standard particle swarm optimization [12, 13]. (iv) Multi-swarm scheme. In (iv), the whole swarm in PSO can be divided into several sub-swarms during the search process so as to explore different sub-regions of the solution space with different search strategies [14, 15]. (v) Hybrid mechanism. As for (v), different evolutionary algorithms (such as genetic algorithm [8], cuckoo search [16], differential evolution [17], simulated annealing [18], artificial bee colony [19], firefly algorithm [20]) and evolutionary operators (like crossover [21], mutation [11]) are integrated together to improve the performance of PSO.

Motivated by the above PSO researches, we put forward a chaos-embedded particle swarm optimization (CEPSO). On the one hand, the chaos-based swarm initialization is applied to yield high-quality initial particles with better stability. On the other hand, the chaotic inertia weight and chaotic sequence based random numbers are introduced into the velocity update scheme for PSO to balance the exploration and exploitation and thus result in a better optimal solution. In addition, two different mutation operators (Gaussian and chaos) are utilized to enhance the swarm diversity and avoid the premature convergence of the CEPSO. Conducted experiments validate that the proposed PSO outperforms several state-of-the-art PSOs regarding their effectiveness and efficiency in the task of numerical function optimization. The rest of this paper is organized as follows. Section 2 introduces the standard PSO. In Sect. 3, the CEPSO is elaborated from three aspects of swarm initialization, parameter selection and mutation strategies adopted in this work, respectively. Experimental results on a set of well-known benchmark functions are reported in Sect. 4. At length, the concluding remarks and future work are provided in Sect. 5.

2 Standard PSO

Particle swarm optimization [1] is a population based meta-heuristic algorithm. The basic principle of PSO mimics the swarm social behaviour such as bird flocking and fish schooling. In PSO, the population is called a swarm and each individual in the swarm is referred as a particle. Each particle in the swarm represents a potential solution to an optimization problem. Specifically, the position of the ith particle can be expressed as a D-dimensional vector $X_i = [x_{i1},x_{i2},\ldots,x_{iD}]$ where $x_{ij} \in [x_{min},x_{max}]$ denotes the position of the jth dimension of the ith particle, and the corresponding velocity can be shown as $V_i = [v_{i1},v_{i2},\ldots,v_{iD}]$ where $v_{ij} \in [v_{min},v_{max}]$ is used to reduce the likelihood of the

particles flying out of the search space. The best previous position (the position giving the best fitness value) of the ith particle is denoted by $pbest_i = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iD})$, while the global best position of the whole swarm found so far is indicated as $gbest = (gbest_1, gbest_2, \ldots, gbest_D)$. To start with, the particles are randomly distributed over the search space with random velocity values. Followed by each particle's velocity is updated using its own previous best experience known as the personal best experience (*pbest*) and the whole swarm's best experience known as the global best experience (*gbest*) until a global optimal solution is found. In PSO, each particle is associated with two properties (velocity vector V and position vector X) and it moves in the search space with a velocity that is dynamically adjusted according to *pbest* and *gbest* simultaneously. Mathematically, velocity and position of particles are updated according to the following formula:

$$v_{ij}(t+1) = \omega \times v_{ij}(t) + c_1 \times r1_{ij} \times [pbest_{ij}(t) - x_{ij}(t)] + c_2 \times r2_{ij} \times [gbest_j(t) - x_{ij}(t)] \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2)$$

where ω is the inertia weight used for balancing the global and local search. In general, a large inertia weight facilitates the global exploration while a small inertia weight tends to facilitate the local exploitation. c_1 and c_2 are positive constants and called the acceleration coefficients reflecting the weight of the stochastic acceleration terms that pull each particle toward $pbest_i$ and *gbest* positions, respectively. $r1_{ij}$ and $r2_{ij}$ denote two random numbers uniformly distributed in the interval [0,1].

Algorithm 1: Pseudocode of the standard PSO algorithm

Input: w_0: inertia weight, c_1, c_2: acceleration factors, N: swarm size, D: swarm dimension.
Process:
1. Randomly initialize the particles of swarm.
2. **while** *number of iterations* or *the stopping criterion* is not met **do**
3. Update the inertia weight.
4. **for** n=1 **to** N **do**
5. Find *pbest* .
6. Find *gbest.*
7. **for** d=1 **to** D **do**
8. Update velocity of particles by Eq.(1)
9. Update position of particles by Eq.(2)
10. **end for**
11. **end for**
12. **end while**

Output: *gbest* particle as the final optimal solution.

3 Proposed CEPSO

3.1 Swarm Initialization

Swarm initialization plays a crucial role in any population based evolutionary algorithms as it affects the convergence speed and quality of the final solution. In general, random initialization is the most frequently used method to generate initial swarm in absence of any information about the solution. From the literature, it can be seen that different initialization strategies have been tested with PSO to improve its performance. Particularly, the chaotic sequence rather than random sequence based initialization is a powerful strategy to diversify the particles of swarm and improve the performance of PSO by preventing the premature convergence [7]. Moreover, it is reported that the stability of PSOs and the quality of final solution can also be improved to some extent [4, 10]. Based on this argument, the commonly used logistic map is employed to generate the initial position instead of the uniform position, which can be described as follows:

$$Ch_{i+1} = \mu \times Ch_i \times (1 - Ch_i), i = 0, 1, 2, \cdots \tag{3}$$

where Ch_i denotes the ith chaotic variable in the interval (0,1), such that the initial $Ch_0 \in (0,1)$ and $Ch_0 \notin (0,0.25,0.5,0.75,1)$. μ is a predetermined constant called bifurcation coefficient. When μ increases from zero, the dynamic system generated by Eq. (3) changes from one fixed-point to two, three,…, and until 2^i. During this process, a large number of multiple periodic components will locate in narrower and narrower intervals of μ as it increases. This phenomenon is obviously free from constraint. But μ has a limit value $\mu_t = 3.569945672$. Note that when μ approaches the μ_t, the period will become infinite or even non-periodic. At this time, the whole system evolves into the chaotic state (behavior). On the other hand, when μ is greater than 4, the whole system becomes unstable. Hence the range $[\mu_t,4]$ is considered as the chaotic region of the whole system. Its bifurcation diagram is illustrated in Fig. 1.

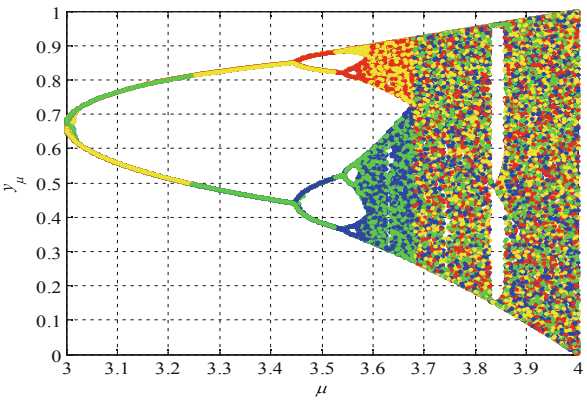


Fig. 1. Bifurcation diagram of logistic map

As described above, the pseudocode of logistic map can be described as below, which is able to generate chaotic sequence and avoid plunging into the small periodic cycles effectively.

Algorithm 2: Pseudocode of logistic map for chaotic sequence

Input: μ: bifurcation coefficient.
Process:
1. Randomly initialize chaotic variables.
2. **while** *number of maximal iterations* is not met **do**
3. **if** chaotic variable plunges into fixed points or small periodic cycles **do**
4. Implement a very small positive random perturbation.
5. Map them by Eq.(3).
6. **else**
7. Update chaotic variables by Eq.(3) directly.
8. **end if**
9. **end while**

Output: the generated chaotic variables.

3.2 Parameter Selection

Proper selection of PSO parameters can significantly affect the performance of particle swarm optimization. It is generally believed that a larger inertia weight facilitates global exploration while a smaller inertia weight tends to facilitate local exploitation to fine-tune the current search space. By changing the inertia weight dynamically, the search capability of PSO can be dynamically adjusted. This is a general statement about the impact of w on PSO's search behavior shared by many other researchers. Based on the nature of ergodicity and non-repetition of the chaos mentioned in Sect. 3.1, the chaotic inertia weight [22] is applied in this work to strike a better balance between the exploration and exploitation, which is defined by adding a chaotic term to the linearly decreasing inertia weight as follows:

$$w(t) = (w_{\max} - w_{\min}) \times \frac{t_{\max} - t}{t_{\max}} + w_{\min} \times ch \tag{4}$$

where $ch \in (0,1)$ is a chaotic number, w_{max} and w_{min} denote the initial value and final value of inertia weight respectively. t is the current iteration of the algorithm and t_{max} is the maximum number of iterations the PSO is allowed to continue.

Note that the random numbers r_1 and r_2 are the key components affecting the convergence behavior of PSO. It is reported that the convergence of PSO is strongly connected to its stochastic nature and PSO uses random sequence for its parameters during a run. In particular, it has been shown that the final optimization results may be very close but not equal when different random sequences are used during the PSO search process [23]. Thus the chaotic sequence generated by Eq. (3) is used to substitute the random numbers r_1 and r_2 in the velocity update equation (the value of Ch varies between 0 and 1) [2].

$$v_{ij}(t+1) = \omega \times v_{ij}(t) + c_1 \times Ch \times [pbest_{ij}(t) - x_{ij}(t)] + c_2 \times (1 - Ch) \times [gbest_j(t) - x_{ij}(t)] \tag{5}$$

3.3 Mutation Strategy

Mutation strategy is an important part of evolutionary computation technique, which can effectively prevent the loss of population diversity and allow a greater region of the search space to be covered. Based on this fact, different mutation operators, such as wavelet [4], gaussian [11], cauchy [11] and levy [24], have been widely used in evolutionary computation, especially for PSO to enhance its global search ability. However, these mutation techniques are not suitable for all kinds of problems, that is to say, most of them are problem-oriented. Thus the two most commonly used mutation strategies, gaussian and chaos, are alternately exploited to mutate the *pbest* and *gbest* based on the swarm diversity defined below, which is expected to well guide the search process of the particle swarm.

$$Div(t) = \frac{1}{N} \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{D} (x_{ij}(t) - \overline{x_j(t)})^2} \tag{6}$$

$$\overline{x_j(t)} = \frac{1}{N} \sum_{i=1}^{N} x_{ij}(t) \tag{7}$$

where N is the swarm size, D denotes the space dimension, $x_{ij}(t)$ is the jth value of ith particle in the tth iteration and $\overline{x_j(t)}$ is the average value of the jth dimension over all the particles in the swarm.

So far, the procedure of CEPSO can be succinctly described as follows.

Algorithm 3: Pseudocode of the proposed CEPSO algorithm

Input: c_1=c_2=2, $Div(0)$=0, swarm size N, swarm dimension D, the threshold of swarm diversity Div_{th}.
Process:
1. Initialize the particles of swarm by **Algorithm 2**.
2. **while** *maximum number of iterations* ($t \le t_{max}$) is not met **do**
3. Calculate w by Eq.(4).
4. Calculate r_1 and r_2 by Eq.(3).
5. **for** n=1 **to** N **do**
6. Find *pbest* .
7. Find *gbest.*
8. **for** d=1 **to** D **do**
9. Update velocity of the particles by Eq.(5).
10. Update position of the particles by Eq.(2).
11. **end for**
12. **end for**
13. Calculate $Div(t)$ according to Eq.(6).
14. **if** ($Div(t) \le Div_{th}$) **do**
15. Update *pbest* and *gbest* by Gaussian mutation.
16. **else**
17. Update *pbest* and *gbest* by Chaos mutation.
18. **end if**
19. **end while**

Output: *gbest* particle as the final optimal solution.

4 Experimental Results and Discussion

To evaluate the performance of the proposed CEPSO, extensive experiments are conducted on a set of well-known benchmark functions consisting of four global optimization problems. Particularly, the performance of PSO with different swarm initialization and different inertia weights are completely compared and investigated respectively. Note that all the test functions are to be minimized, they are numbered f_1–f_4 given in Table 1, including their expression, dimension, allowable search space, global optimum and property, respectively.

Table 1. Dimensions, search ranges, global optimum values and properties of test functions

Test functions	Dimensions (n)	Search range	Global optimum	Properties
Sphere (f_1)	10/30	$[-10,10]^n$	0	Unimodal
Schwefel (f_2)	10/30	$[-100,100]^n$	0	Unimodal
Rastrigin (f_3)	10/30	$[-5.12,5.12]^n$	0	Multimodal
Ackley (f_4)	10/30	$[-32,32]^n$	0	Multimodal

(1) Sphere function

$$\min f_1(x) = \sum_{i=1}^{n} x_i^2$$

where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-10 \le x_i \le 10$.

(2) Schwefel function

$$\min f_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$$

where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-100 \le x_i \le 100$.

(3) Rastrigin function

$$\min f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$$

where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-5.12 \le x_i \le 5.12$.

(4) Ackley function

$$\min f_4(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

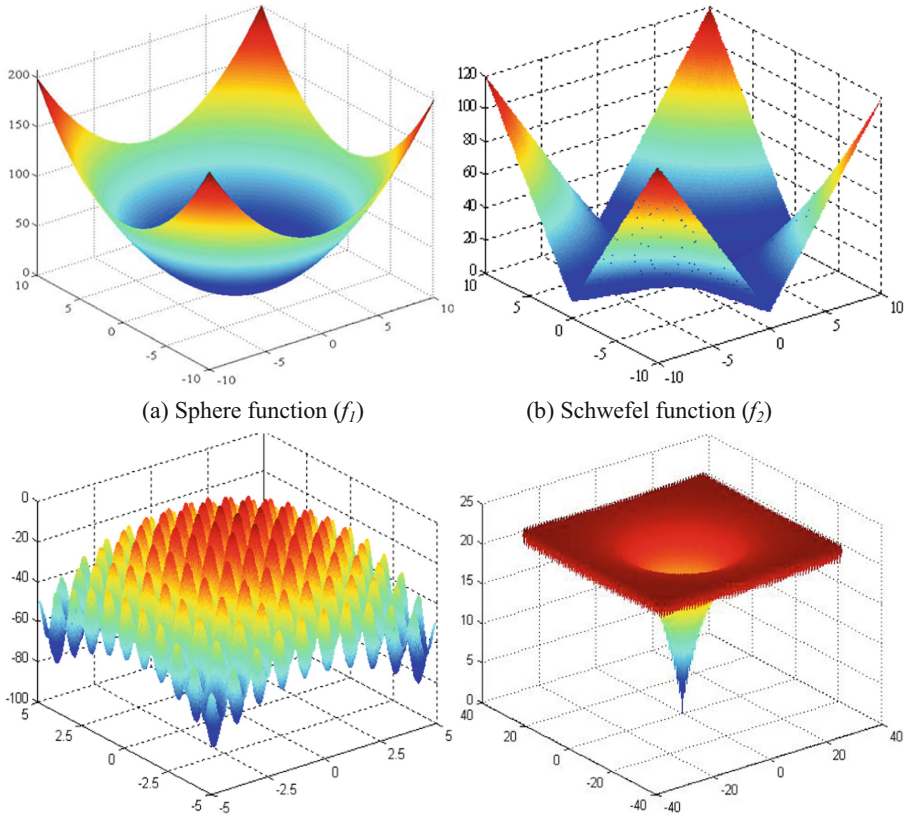
where the global optimum $x^* = 0$ and $f(x^*) = 0$ for $-32 \le x_i \le 32$.

Figure 2 depicts the 2-dimensional graphical shows of the four well-known benchmark functions.

To illustrate the effect of different initialization methods, inertia weights and mutation strategies mentioned above, different combinations of PSO with random/chaos swarm initialization, standard/chaotic parameters, and without/chaos mutation are tested respectively. For readability, note that different PSO paradigms are specified in Table 2 by their acronyms, respectively.

Consider the following given conditions: the swarm size $N = 40$, the standard parameters include $w_{max} = 0.9$, $w_{min} = 0.4$ and the acceleration coefficients $c_1 = c_2 = 2$ for all the related PSO variants. The threshold of the swarm diversity is predetermined $Div_{th} = 2.8\text{e}{-20}$ by trial and error. For each test function, 30 independent runs are performed by each PSO, and each run is with 1000 iterations. The algorithm terminates when it reaches the maximum number of allowed iterations.

From Table 3, it can be clearly observed that CEPSO evidently outperforms other PSO variants on all the four test functions. Taking f_1 and f_2 for example, note that under the same conditions of the swarm initialization and related parameters, the PSOs with mutation strategies markedly surpass those without mutations. This validates the importance of mutation operators to sustain the swarm diversity without being trapped in local optima. In particular, it is worth noting that the performance of PSO with chaos



(a) Sphere function (f_1) (b) Schwefel function (f_2)

(c) Rastrigin function (f_3) (d) Ackley function (f_4)

Fig. 2. Graphical shows of the benchmark test functions

Table 2. Acronyms of different PSO variants

Initialization	Parameters	Mutation	Acronym
Random	Standard	Without	PSO-Rnd-S-W
		Gaussian/chaos	PSO-Rnd-S-M
	Chaos-based	Without	PSO-Rnd-C-W
		Gaussian/chaos	PSO-Rnd-C-M
Chaos	Standard	Without	PSO-Chs-S-W
		Gaussian/chaos	PSO-Chs-S-M
	Chaos-based	Without	PSO-Chs-C-W
		Gaussian/chaos	PSO-Chs-C-M

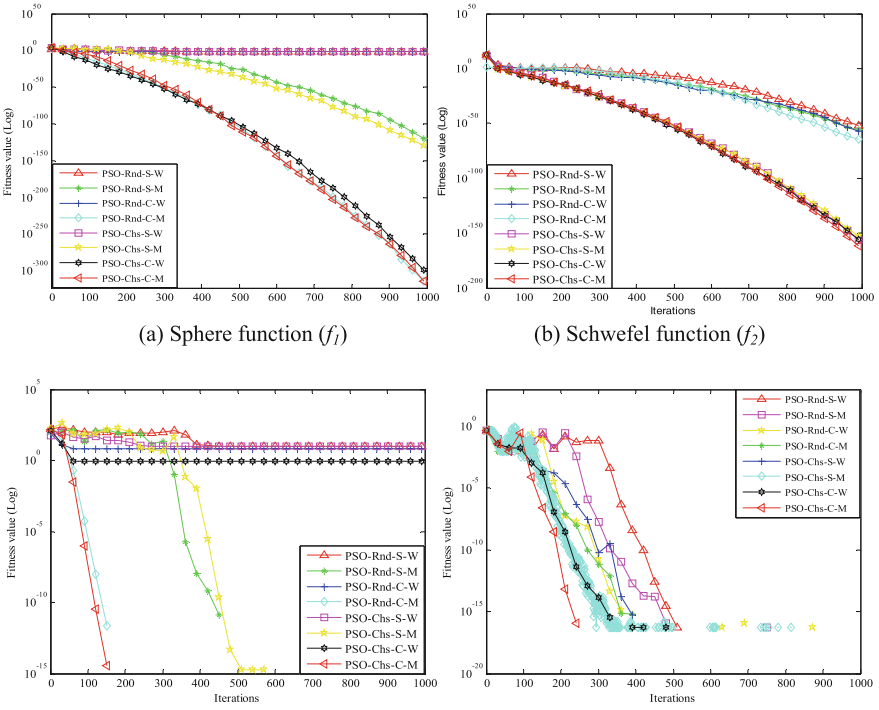
Table 3. Results of PSO with different configurations ($d = 10$)

Functions	PSO algorithm	Best solution	Average solution	Standard deviation
Sphere	PSO-Rnd-S-W	5.1001e−003	5.8303e−002	5.1901e−002
	PSO-Rnd-S-M	2.7594e−126	2.4107e−119	4.1704e−119
	PSO-Rnd-C-W	1.3901e−002	9.0386e−002	7.0010e−002
	PSO-Rnd-C-M	8.2914e−320	6.0543e−313	0.0000e−000
	PSO-Chs-S-W	1.1621e−002	3.9400e−002	2.5100e−002
	PSO-Chs-S-M	1.6636e−132	1.5062e−119	2.1290e−119
	PSO-Chs-C-W	1.1666e−315	3.2793e−298	0.0000e−000
	PSO-Chs-C-M	6.3734e−321	1.4005e−314	0.0000e−000
Schwefel	PSO-Rnd-S-W	1.5868e−053	4.7589e−053	2.2430e−053
	PSO-Rnd-S-M	1.2530e−063	1.4764e−063	1.3052e−063
	PSO-Rnd-C-W	2.0361e−155	2.1141e−155	2.1141e−155
	PSO-Rnd-C-M	4.7263e−160	3.7903e−158	5.1224e−158
	PSO-Chs-S-W	3.5786e−057	3.9307e−057	2.7731e−057
	PSO-Chs-S-M	3.4395e−065	3.5180e−058	4.9752e−058
	PSO-Chs-C-W	2.7406e−159	5.2705e−159	5.6281e−159
	PSO-Chs-C-M	1.6481e−165	2.5707e−161	3.5564e−161
Rastrigin	PSO-Rnd-S-W	1.0499e+001	1.1238e+001	6.5312e−001
	PSO-Rnd-S-M	0.0000e−000	0.0000e−000	0.0000e−000
	PSO-Rnd-C-W	6.4490e−000	1.0079e+001	2.9940e−000
	PSO-Rnd-C-M	0.0000e−000	0.0000e−000	0.0000e−000
	PSO-Chs-S-W	9.7562e−000	1.4498e+001	3.2530e−000
	PSO-Chs-S-M	0.0000e−000	0.0000e−000	0.0000e−000
	PSO-Chs-C-W	8.8020e−001	6.2974e−000	3.8701e−000
	PSO-Chs-C-M	0.0000e−000	0.0000e−000	0.0000e−000
Ackley	PSO-Rnd-S-W	8.3506e−016	7.6226e−007	7.9614e−007
	PSO-Rnd-S-M	5.8233e−017	7.5088e−007	5.2016e−006
	PSO-Rnd-C-W	7.6952e−016	6.6319e−007	3.0892e−008
	PSO-Rnd-C-M	5.5511e−017	6.9783e−007	7.6836e−007
	PSO-Chs-S-W	7.6952e−016	7.1015e−007	7.6836e−007
	PSO-Chs-S-M	4.8102e−017	6.8769e−009	7.6836e−008
	PSO-Chs-C-W	5.3357e−016	8.3361e−008	5.4387e−008
	PSO-Chs-C-M	9.2618e−018	5.1679e−010	2.3911e−010

swarm initialization is far superior to those corresponding PSOs with random initialization except for PSO-Chs-S-W and PSO-Rnd-S-W for function f_1, which implies that PSO with the chaos-based initialization can alleviate its inherent defects of low stability. In other words, just as the conclusions drawn in references [4, 7], this fully demonstrates the significance of the high-quality initial particles to the convergent performance of PSO algorithm. In addition, note also that PSO with the chaotic parameters can achieve better performance than those with the standard parameters under the same other experimental conditions. As for the test function f_3, due to it owns a large number of local optima, thus it is difficult to find the global optimization values. However, it is interesting to note that its optimal value can be found based on the PSO-Rnd-S-M, PSO-Rnd-C-M, PSO-Chs-S-M and PSO-Chs-C-M respectively so as to further show the importance of mutation operation. Likewise, CEPSO is able to get better solutions for function f_4 even though it has one narrow global optimum basin and many minor local optima. In sum, the promising results obtained by the CEPSO are largely attributed to the chaos-based swarm initialization, the chaotic inertia weight and chaotic sequence based random numbers as well as two different mutation strategies (chaos and levy), all of these are complementary to each other and the appropriate combination makes them benefit from each other.

To illustrate the particles search process, Fig. 3 depicts the evolution curves of PSO with different configurations for all the four test functions, which clearly shows that PSO-Chs-C-M performs much better than the other PSO variants in almost all cases. To be specific, taking Fig. 3(a) for example, the evolution curve of PSO-Chs-C-M consistently decreases fastly compared to that of PSO-Rnd-S-W, PSO-Rnd-C-W and PSO-Chs-S-W respectively. In contrast, the convergence rates of PSO-Rnd-S-M and PSO-Chs-S-M are not as fast as that of PSO-Chs-C-M, PSO-Chs-C-W and PSO-Rnd-C-M, both of them evolve slowly as the search proceeds but are better than the PSO-Rnd-S-W, PSO-Rnd-C-W and PSO-Chs-S-W respectively. On the other hand, it is interesting to note that PSO-Chs-C-M yields the fastest convergence rate at the early 150 iterations in Fig. 3(c). Compared to PSO-Rnd-S-W, PSO-Rnd-C-W, PSO-Chs-S-W and PSO-Chs-C-W, the other three PSOs with mutation also evolve fast in the early 500 iterations and finally converge to the global optima.

To further validate the effectiveness of the proposed algorithm, CEPSO is compared with the other seven PSO variants in Table 4, including LPSO [25], FIPSO [26], HPSO-TVAC [27], DMSPSO [28], CLPSO [29], LFPSO [30] and OPSO [31]. Note that the dimension of all the functions is set as 30 in this section. The mean best solution (Mean) and standard deviation (Std.) are applied to measure the performance. From the results shown in Table 4, we can see that CEPSO gets rank 1 three times in spite of the rank 3 on function f_4. According to the final rank, it is clearly observed that CEPSO can achieve better performance than the others in terms of the average best solution and standard deviation. To summarize, the encouraging results obtained by the CEPSO are largely ascribed to the chaos-based swarm initialization, chaotic parameters and multi-mutation strategies exploited in this work, respectively.



(a) Sphere function (f_1) (b) Schwefel function (f_2)

(c) Rastrigin function (f_3) (d) Ackley function (f_4)

Fig. 3. Evolution curves of PSO with different configurations

Table 4. Comparison of CEPSO with other seven PSO variants ($d = 30$)

Func	Item	CLPSO	HPSO-TVAC	FIPSO	LPSO	DMSPSO	LFPSO	OPSO	CEPSO
f_1	Mean	1.58e−12	2.83e−33	2.42e−13	3.34e−14	2.65e−31	4.69e−31	6.45e−18	**8.06e−106**
	Std.	7.70e−13	3.19e−33	1.73e−13	5.39e−14	6.25e−31	2.50e−30	4.64e−18	**3.58e−102**
	Rank	8	2	7	6	3	4	5	1
f_2	Avg.	2.51e−08	9.03e−20	2.76e−08	1.70e−10	1.57e−18	2.64e−17	1.26e−10	**2.62e−49**
	Std.	5.84e−09	9.58e−20	9.04e−09	1.39e−10	3.79e−18	6.92e−17	5.58e−11	**5.11e−50**
	Rank	7	2	8	6	3	4	5	1
f_3	Avg.	9.09e−05	9.43e−00	6.51e+01	3.51e+01	2.72e+01	4.54e−00	6.97e−00	**0**
	Std.	1.25e−04	3.48e−00	1.34e+01	6.89e−00	6.02e−00	1.03e−01	3.07e−00	**0**
	Rank	2	5	8	7	6	3	4	1
f_4	Avg.	3.66e−07	7.29e−14	2.33e−07	8.20e−08	1.84e−14	**1.68e−14**	6.23e−09	6.36e−14
	Std.	7.57e−08	3.00e−14	7.19e−08	6.73e−08	4.35e−15	**4.84e−15**	1.87e−09	8.09e−13
	Rank	8	4	7	6	2	1	5	3
Avg. rank		6.25	3.25	7.50	6.25	3.50	3.00	4.75	1.50
Final rank		6	3	7	6	4	2	5	1

5 Conclusions and Future Work

In this paper, we propose a chaos-embedded particle swarm optimization algorithm (CEPSO). On the one hand, the chaos-based swarm initialization is first used to yield high-quality initial particles with better stability. On the other hand, the chaotic inertia weight and the chaotic sequence based random numbers are introduced into the velocity update scheme for PSO to improve its global and local search ability. In the meanwhile, two different mutation operators (chaos and levy) are used to enhance the swarm diversity and avoid the premature convergence. At length, extensive experiments on a set of well-known benchmark functions demonstrate that CEPSO is much more effective than several other PSOs in dealing with numerical function optimization.

As future work, we plan to compare CEPSO with more state-of-the-art PSO variants in the task of complex multi-optima and multi-objective problems, or even some real-world applications from other fields to further investigate the effectiveness of the CEPSO. More interesting future work is to introduce the other most common chaotic maps, viz. Tent map, Lozi map, Arnold's cat map, Sinai map, Burgers map, Dissipative standard map, Tinkerbell map, Circle map and Sinusoidal map, into the PSO to investigate how to improve its performance without being trapped in local optima. Meanwhile, we also intend to delve deeper into the parallelization of CEPSO for large-scale optimization problems and exploring the use of different chaotic parameters for PSO in different scenarios simultaneously, especially for the adequate parameter tuning in a wide range of problems. Lastly, and arguably most importantly, the qualitative relationships between the chaos-based swarm initialization and the stability of PSO, from the viewpoint of mathematics, will be elaborated and proved comprehensively.

Acknowledgment. This work is fully supported by the Program of the Science and Technology Department of Xinjiang Uygur Autonomous Region (No. 2022D01A16) and the Program of the Applied Technology Research and Development of Kashi Prefecture (No. KS2021026).

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization In: IEEE Conference on Neural Networks (ICNN 1995), pp. 1942–1948 (1995)
2. Bharti, K., Singh, P.: Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering. Appl. Soft Comput. **43**, 20–34 (2016)
3. Chuang, L., Hsiao, C., Yang, C.: Chaotic particle swarm optimization for data clustering. Expert Syst. Appl. **38**(12), 14555–14563 (2011)
4. Tian, D., Shi, Z.: MPSO: Modified particle swarm optimization and its applications. Swarm Evol. Comput. **41**, 49–68 (2018)
5. Petrović, M., Vukovic, N., Mitic, M., et al.: Integration of process planning and scheduling using chaotic particle swarm optimization algorithm. Expert Syst. Appl. **64**, 569–588 (2016)
6. Das, P., Behera, H., Panigrahi, B.: A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. Swarm Evol. Comput. **28**, 14–28 (2016)
7. Tian, D.: Particle swarm optimization with chaos-based initialization for numerical optimization. Intell. Autom. Soft Comput. **24**(2), 331–342 (2018)

8. Dong, N., Wu, C., Ip, W., et al.: An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection. Comput. Math. Appl. **64**, 1886–1902 (2012)
9. Xue, B., Zhang, M., Browne, W.: Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms. Appl. Soft Comput. **18**, 261–276 (2014)
10. Tian, D., Zhao, X., Shi, Z.: Chaotic particle swarm optimization with sigmoid-based acceleration coefficients for numerical function optimization. Swarm Evol. Comput. **51**, 126–145 (2019)
11. Tian, D., Zhao, X., Shi, Z.: DMPSO: Diversity-guided multi-mutation particle swarm optimizer. IEEE Access **7**(1), 124008–124025 (2019)
12. Beheshti, Z., Shamsuddin, S.: Non-parametric particle swarm optimization for global optimization. Appl. Soft Comput. **28**, 345–359 (2015)
13. Liu, Z., Ji, X., Liu, Y.: Hybrid non-parametric particle swarm optimization and its stability analysis. Expert Syst. Appl. **92**, 256–275 (2018)
14. Chen, Y., Li, L., Peng, H., et al.: Dynamic multi-swarm differential learning particle swarm optimizer. Swarm Evol. Comput. **39**, 209–221 (2018)
15. Xia, X., Gui, L., Zhan, Z.: A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting. Appl. Soft Comput. **67**, 126–140 (2018)
16. Bouyer, A., Hatamlou, A.: An efficient hybrid clustering method based on improved cuckoo optimization and modified particle swarm optimization algorithms. Appl. Soft Comput. **67**, 172–182 (2018)
17. Mao, B., Xie, Z., Wang, Y., et al.: A hybrid differential evolution and particle swarm optimization algorithm for numerical kinematics solution of remote maintenance manipulators. Fusion Eng. Des. **124**, 587–590 (2017)
18. Javidrad, F., Nazari, M.: A new hybrid particle swarm and simulated annealing stochastic optimization method. Appl. Soft Comput. **60**, 634–654 (2017)
19. Li, Z., Wang, W., Yan, Y., et al.: PS-ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. Expert Syst. Appl. **42**(22), 8881–8895 (2015)
20. Aydilek, İ: A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. Appl. Soft Comput. **66**, 232–249 (2018)
21. Meng, A., Li, Z., Yin, H., et al.: Accelerating particle swarm optimization using crisscross search. Inf. Sci. **329**, 52–72 (2016)
22. Feng, Y., Yao, Y., Wang, A.: Comparing with chaotic inertia weights in particle swarm optimization. In: IEEE Conference on Machine Learning and Cybernetics (ICMLC 2007), pp. 329–333 (2007)
23. Alatas, B., Akin, E., Ozer, A.: Chaos embedded particle swarm optimization algorithms. Chaos Solitons Fractals **40**(4), 1715–1734 (2009)
24. Wang, H., Wang, W., Wu, Z.: Particle swarm optimization with adaptive mutation for multimodal optimization. Appl. Math. Comput. **221**, 296–305 (2013)
25. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: IEEE Congress on Evolutionary Computation (CEC 2002), pp. 1671–1676 (2002)
26. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. IEEE Trans. Evol. Comput. **8**(3), 204–210 (2004)
27. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans. Evol. Comput. **8**(3), 240–255 (2004)
28. Liang, J., Suganthan, P.: Dynamic multi-swarm particle swarm optimizer. In: IEEE Conference on Swarm Intelligence Symposium (SIS 2005), pp. 124–129 (2005)

29. Liang, J., Qin, A., Suganthan, P., et al.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. **10**(3), 281–295 (2006)
30. Haklı, H., Uğuz, H.: A novel particle swarm optimization algorithm with Levy flight. Appl. Soft Comput. **23**, 333–345 (2014)
31. Ho, S., Lin, H., Liauh, W., et al.: OPSO: Orthogonal particle swarm optimization and its application to task assignment problems. IEEE Trans. Syst. Man Cybern. A Syst. Hum. **38**(2), 288–298 (2008)